

Design Tools and Methodologies for the FPGA Based ROD

Atlas - LBNL/Wisconsin Group

K. Dao, D. Fasching, O. Hayes, R. Jared
K Marks, M Nagel

March 20, 1999

Issues Facing Large Designs Like the ROD

- Advanced CAD tools and better design methodologies are needed for complex electronics systems:
 - FPGA densities are increasing while costs drop.
 - 100K to 1M gate FPGAs available today.
 - Each engineer must tackle bigger and more complicated designs to fully utilize these FPGAs.
 - Cost and Development time cannot increase with FPGA density.
 - Testing and Debugging becomes very difficult.
 - Embedded microprocessors also face similar issues.

Key Features of the ROD Design Methodology

- Mentor CAD tools to unify the design environment from the FPGA gate level to the full board level.
- Software based approach to logic design.
 - Write commented, structured code instead of drawing pictures.
 - Work at a higher level of abstraction to allow the designer to focus on the functionality rather than the implementation.
- Computer simulations to cut debugging time in the lab.
- Component libraries to reduce pinout/functional errors.
- Board level simulation to reduce timing and layout errors.
- Create a behavioral model to test algorithmic logic.

Mentor Design Environment

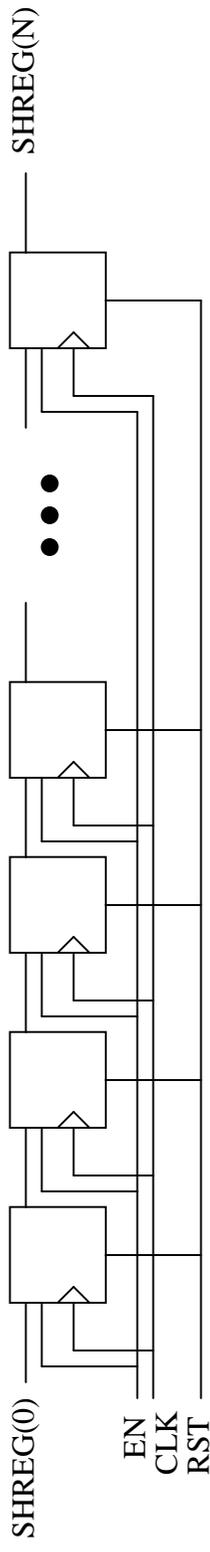
- Using a common design environment that supports multi-user access helps people work together on the same board
 - Access control locks so only one user can be editing a file.
 - Supports simulation at the FPGA level and board level to check that all the pieces work together.
 - Reduces errors introduced at various phases in the design.
 - Layout is cross checked with the circuit schematics which are further checked by the board level simulation.
 - The software and the necessary support staff are expensive.
 - LBNL has already made the investment in establishing a stable CAD software environment from its work in past projects (i.e. Babar Trigger, PLL).

Hardware Description Languages

- HDLs such as VHDL provide a programming model for logic design.
 - HDLs specify the behavior of a design in an unambiguous, mathematical way.
 - Structured, commented code is easier to read and understand.
 - Higher level of abstraction- don't think in terms of drawing gates.
 - Logic can be developed faster at the HDL level.
 - Designs are portable to other FPGAs or ASICs.
 - Modules of code can be parameterized and later reused.
 - High degree of debugging support when simulation is used:
 - Compiler syntax checking, breakpoints, step through HDL code as it executes, print statements, and file I/O.

HDL vs Schematic Capture Example

- Shift register with shift enable and async reset



```
Shifter: Process (clk, rst)
Begin
  if (rst = '1') then
    shreg <= (others => '0');
  elsif (clk'event and clk = '1') then
    for i in 1 to n do
      shreg(i) <= shreg(i-1);
    end loop;
  end if;
end process;
```

Using Simulation is Essential

- Careful design alone cannot prevent all errors from occurring.
 - In a complex board with multiple designers, there will be errors.
 - Finding errors at the test bench is too slow (~ few errors/day).
 - Severe errors may require another spin of the board at great cost.
- Good design practices are still required.
 - Cannot rely on simulation to fix problems.
 - Beware of designing by trial and error .
 - Spending time to write good code in the first place is an even faster way to fix bugs.
- **ASIC design axiom: Anything not simulated will not work.**

Simulation Methodology

- Test low level HDL modules with simple test patterns.
- Move on to bigger patterns at higher levels in the design.
- Full board level simulation takes more CPU time and bugs become increasingly difficult to find and correct.
 - Simulate at low levels to find the obvious bugs first.
 - Use component libraries to model the behavior of support components and ensure the correct pinouts are used for layout.
- Once the functionality has been verified, add in worst case component delays to verify overall timing and performance.
 - FPGA timing can be back-annotated into the board simulation.

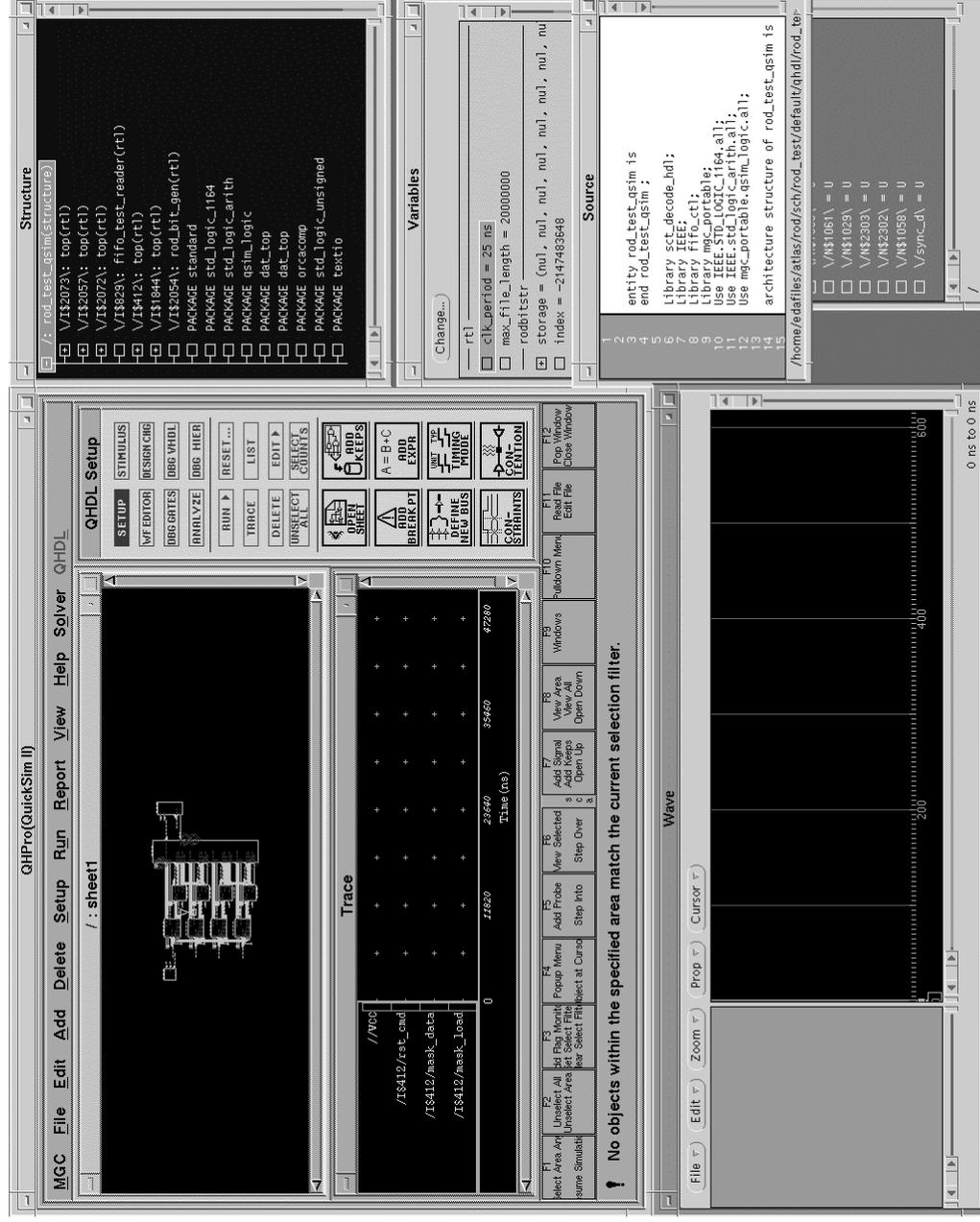
Testing and Verification Issues

- Use a behavioral model to generate test patterns and view expected results at different stages in the hardware.
 - Create the behavioral model in Fortran, C, Pascal, etc.
 - These behavioral models can also help evaluate various hardware architectures when designing the board.
 - i.e. FIFO depth, trigger algorithm efficiency, hardware performance.
- **Use test suites to automate tests.**
 - Quickly recheck the entire design after any logic changes.
 - The same test vectors can be loaded into the ROD input memories for production testing.

Debug / Diagnostic Issues

- Debugging at the test bench will still be needed.
 - Simulation cannot catch some errors - race conditions, transmission line problems, power distribution, cooling, etc.
 - FPGA internal signals can be routed to spare I/O pins so internal signals can be monitored during runtime.
 - A via per solder ball provides access to all FPGA BGA pins.
- FPGAs will have spare lines connecting each other in case additional communications paths are needed.
- ROD will use a fully synchronous design (except with DSPs and VME bus) to simplify board timing.
- Input Memories and DSPs can play/record test patterns.
 - Gatherer output memory can also be loaded by VME to play a test pattern.

Mentor Graphics Simulation Environment



DSP Design Issues

- Different model of execution than FPGAs.
- Software simulation is similar to hardware simulation.
 - Step through instructions and set breakpoints.
 - Monitor variables
- Debugging at the test bench.
 - No operating system to help debug the software.
 - Must use breakpoints to trap errors and stop the processor.
 - Monitor the value of any variable or register.
 - Read/Write to the processor's internal/external memory.
 - Difficult to monitor the operation of the DSPs in real time.
- A behavioral I/O model of the DSPs will be used to help simulate the interface logic between FPGAs and DSPs.

Conclusions

- Advances in FPGA / DSP / CAD Software technology allow us to build a very complex board like the ROD.
 - Engineering resources, cost, and time are limited.
- Advanced CAD tools provide the framework for designing and testing the entire board in the computer.
- HDLs provide a high-level programming paradigm.
- Simulation is an essential part of the design process.
 - Boards need to be tested in the computer not in the lab.
- CAD software is a tool to help manage the design, not a replacement for a good engineer.